ECE 445

Fall 2023

Final Report

Smart Availability Indicator for ECEB Study Rooms

Siddarth Iyer, Ritvik Goradia

TA: Sanjana Pingali

Group 27

Abstract

This project aims to build a device that could be placed in study rooms to let students know if the room is occupied or not. We have successfully built this product that uses a motion sensor to detect the presence of users in the room and relays this information to an online database that students can access from their phones anywhere in the world. This document details the motivation, design, outcome, and cost of the product created.

Table of Contents

1. Introduction

1.1 Problem

The Electrical and Computer Engineering Building (ECEB) offers a limited number of study and discussion rooms across its five floors, which prove invaluable for students aiming to collaborate on assignments, prepare for exams, or engage in general study sessions. There is currently no system in place to identify vacant rooms and as a result, each day students tirelessly go back and forth, scouring each floor in hopes of finding an empty study room. Worse still, there are also many students who travel all the way to the ECEB only to discover that all rooms are occupied forcing them to then seek alternatives in different buildings.

1.2 Solution

Our proposal presents a solution aimed at enabling students to easily and conveniently check room availability via a website. We plan to implement a system that will replace the existing motion sensor and automation infrastructure. Our proposed system not only retains the current capabilities of automatically controlling room lighting based on occupancy and adjusting brightness levels but will also seamlessly transmit real-time occupancy data to a central server using Wi-Fi, allowing students to conveniently view room availability information from their smartphones or laptops on a website anytime/anywhere.

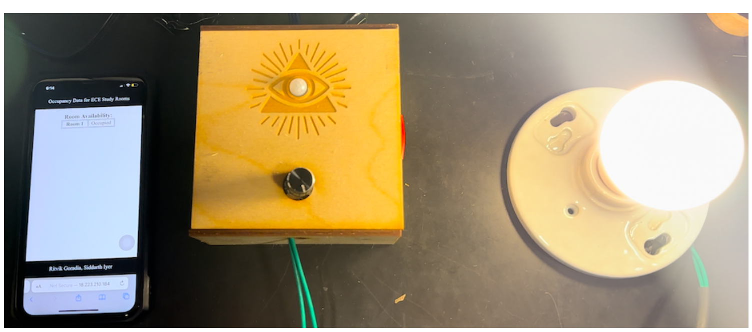

Figure 1: Final product along with website shown on phone

1.3 Visual Aid

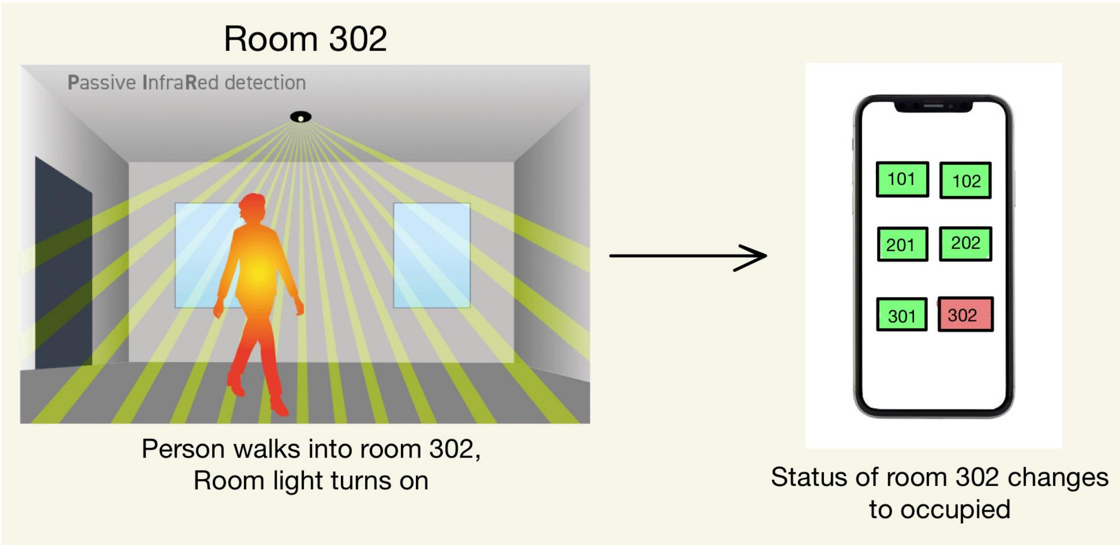

Figure 2: High-level illustration of our project [1][2]

1.4 High-Level requirements

- The lighting system should activate within three seconds upon detecting the presence of a user in the room. The light should turn off automatically after a five-minute period when the user leaves the room. The lights activating act as a signal that the room is occupied. This is also a current feature of the study rooms that we want to replicate.
- Users should be able to adjust the light's brightness from dim to its maximum intensity using a dial/slider. (Demonstrable by displaying three distinct brightness levels: dim, intermediate, and bright.) This requirement is also a current feature of the study rooms that we want to replicate. Different students have different brightness preferences and this allows the study rooms to be appealing to more people.
- Any changes in room availability should be reflected on the webpage within 10 seconds. This requirement will allow us to display the room occupancy data to the user in real time.

1.5 Subsystem Overview

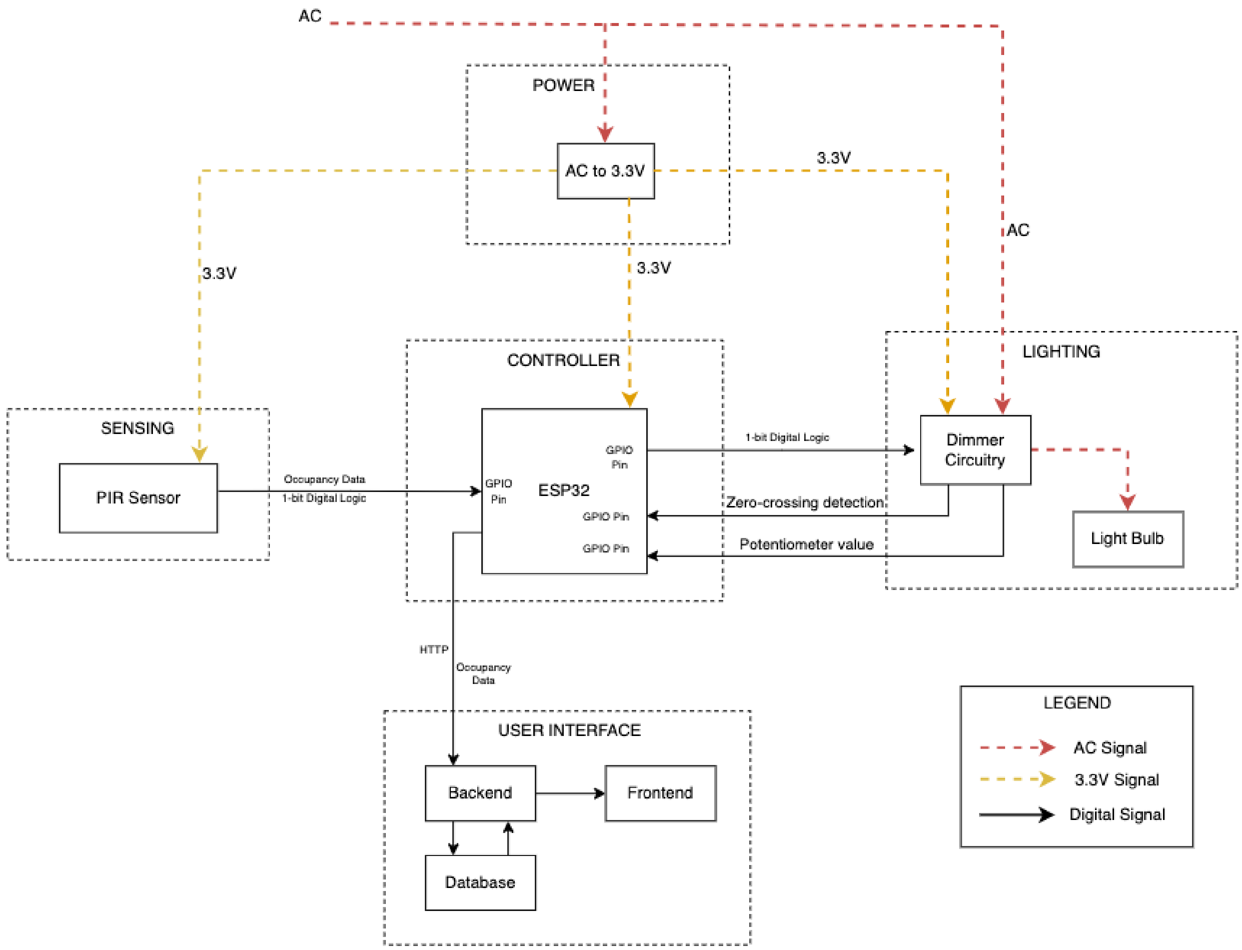

Figure 3: Block diagram depicting interactions between modules

1.5.1 Power

This subsystem powers the hardware subsystems of this project (Sensing, Controller, Lighting). For the prototype, the system will be powered by an AC power outlet. The power subsystem uses an AC to 3.3 V DC converter to create a 3.3 V rail. This rail will be used to power the microcontroller, dimmer circuitry, and motion sensor.

1.5.2 Sensing

The sensing sub-system is responsible for determining room availability. It makes use of a PIR (Passive Infrared) sensor to detect changes in heat energy emitted in the room. When movement is detected, this subsystem sends a pulse to the controller to inform it that there is activity in the room.

1.5.3 Controller

This subsystem acts as the brains of the project and makes use of an ESP32 Microcontroller. It receives data from the sensing module to determine if the room is occupied or not and sends this data to the User Interface subsystem to be displayed on the website. Using a dial, the controller determines how bright the user wants the lights and accordingly sends pulses at a constant rate to dim the lightbulb.

1.5.4 Lighting

This submodule is responsible for turning on/off the lightbulb and allowing the user to adjust its brightness using a potentiometer dial. This circuit receives pulses from the controller and accordingly adjusts the duty cycle of the lightbulb to change its brightness.

1.5.5 User Interface

The User Interface subsystem allows the user to view which rooms are available/occupied from anywhere in the world. This consists of a website running on an AWS (Amazon Web Services) Linux server. It receives the occupancy data sent from the controller and automatically updates the table with that information in real-time.

2. Design

2.1 Power Subsystem

2.1.1 Linear Regulator

The power subsystem is responsible for powering the various subsystems in the project with 3.3 V. For this we make use of a linear regulator. We input a voltage of 9 V from a 9 V battery, and 3.3 V is outputted by the regulator. Since we make use of an adjustable voltage regulator, it requires two external resistors to adjust the output voltage. Based on (1) we chose to use R1 = 200 Ω and R2 = 330 Ω. Figure 4 describes how the resistors are connected to the linear regulator.

$$V_{OUT} = 1.25 \cdot \left(1 + \frac{R2}{R1}\right) \tag{1}$$

To validate that the linear regulator did output 3.3 V we measured the voltage across the output pin of the regulator using a voltmeter and observed that the voltage fell within our required threshold (3.3 V ± 0.3). Shown in Fig. 5, we can see the output of the voltmeter.

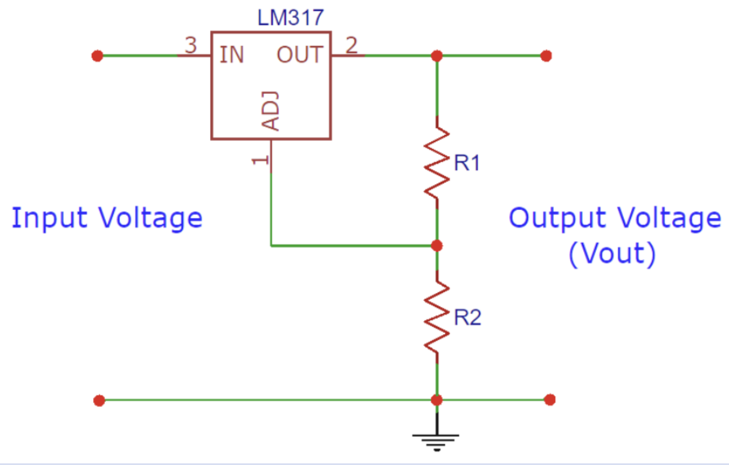

Figure 4: Circuit diagram of a voltage regulator[3]

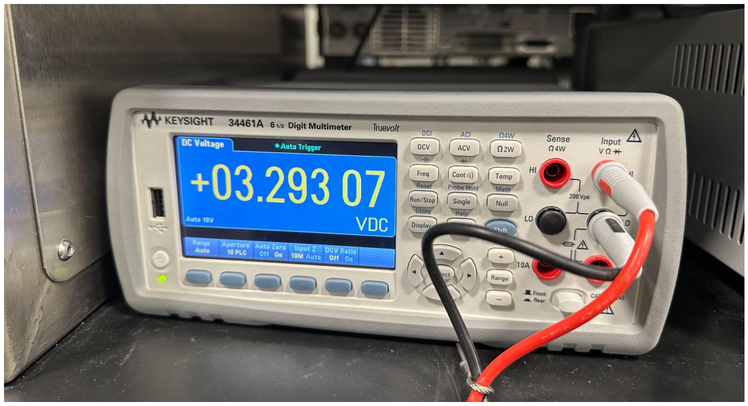

Figure 5: Output of linear regulator

2.1.2 Tolerance Analysis

Required Output: 3.3 V

Battery Headroom: 9 V - 3.3 V = 5.7 V

Dropout Voltage: 0.4 V

Power Dissipation:

$$Iout * (Vin – Vout) = 0.12 * (9 - 3.3) = 0.684 \text{ W} \tag{2}$$

Thermal Resistance:

$$T_{ja} = i_{out} * (V_{in} - V_{out}) * (\Theta_{jc}) \tag{3}$$

$$= 0.12 * 5.7 * 96 = 65.66 \text{ °C}$$

Max Operation Temperature: 110 °C

Therefore, we are within operable limits.

Table 1: Voltage dropouts for linear regulator[4]

V_{DROP}	Dropout Voltage	I_{OUT} = 10mA	—	5	8	mV
		I_{OUT} = 300mA	—	125	200	
		I_{OUT} = 600mA	—	250	400	

2.1.3 Design Issues & Alternatives

We required a 5 V power rail to power the PIR Sensor Module we initially planned on using. We therefore decided to use an AC-DC 5 V converter to create the 5 V power rail, and then make use of a linear regulator to step the 5 V down to 3.3 V. However after switching to the bare PIR Sensor, the 5 V power rail was no longer necessary and we instead decided to use an AC-DC 3.3 V convertor. However this component did not get delivered and as a result, we ended up designing a linear regulator that would be powered by a 9 V battery and would output 3.3 V

2.2 Sensing Subsystem

2.2.1 PIR Sensor

To detect activity in the room, we make use of a PIR Motion Sensor labeled BS-412. This module has two Infrared sensors placed a small distance apart. When there is no movement in the room, both sensors detect the same infrared waves at the same time, and the module outputs a 0 V (digital low) signal. If there is movement in the room, the sensors detect the changes in infrared at slightly different times. When this happens, the module outputs a 3.3 V Digital High pulse to the microcontroller. To confirm the sensor's functionality, we connected the output of the sensor to the oscilloscope. Shown below in Fig. 6 is the waveform seen on the oscilloscope

We initially kept the sensor covered such that it would not detect motion. This corresponds to the initial low signal seen in the wave. We then uncovered the sensor and allowed it to detect motion which resulted in the pulse seen in the wave. We then covered the sensor again, resulting in a low signal towards the end. Once we were confident that the sensor was functioning, we then interfaced it with the microcontroller.

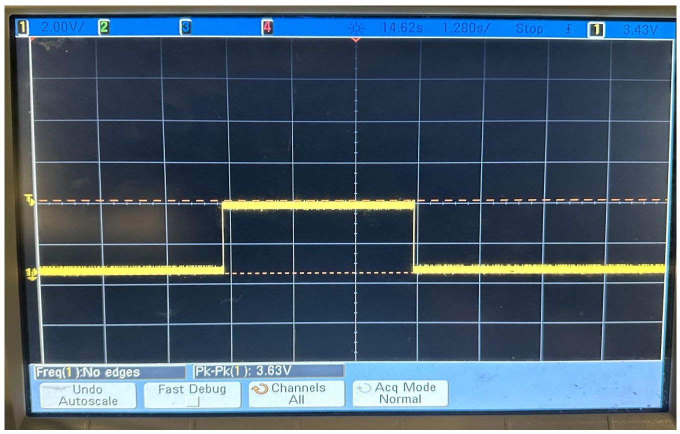

Figure 6: Pulse sent by PIR sensor indicating movement

2.2.2 Design Issues & Alternatives

We initially intended to use a PIR Sensor Module for motion detection which required a 5 V power supply. This module included various peripherals and an on-board processor. Upon further research, we realized that we could offload the processing to our main microcontroller. Consequently, we opted to switch to a bare PIR sensor. This decision not only reduced costs significantly due to the lower cost of the bare sensor but also eliminated the need for a 5 V power rail in our project. This then allowed us to simplify our power submodule leading to further cost reduction while maintaining the intended functionality.

2.3 Controller Subsystem

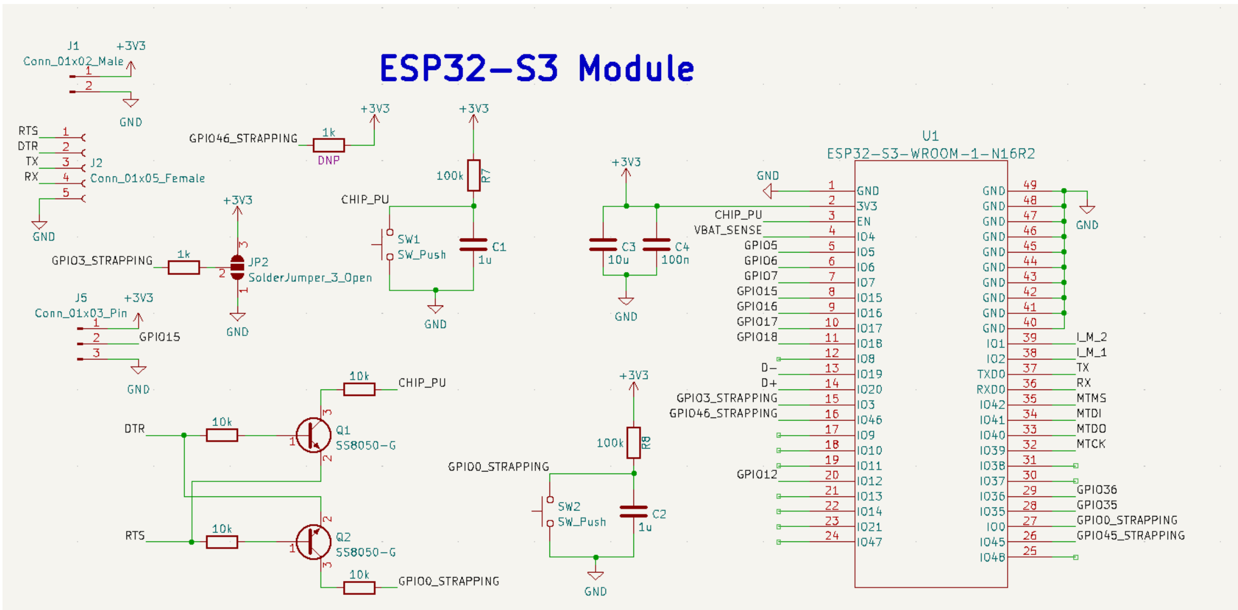

Figure 7: PCB schematic for Controller subsystem

2.3.1 Room Occupancy

The microcontroller we are using is an ESP32-S3 WROOM due to its WiFi capabilities. The microcontroller receives occupancy data via a 1-bit digital logic output from the sensing module. This data is subsequently passed on to the AWS server via HTTP (Hypertext Transfer Protocol). The data sent by the controller is a tuple that looks like: [Room Number, Status (0/1)].

2.3.2 Dimmer

The second function of the controller is to read the brightness level indicated by the potentiometer dial and accordingly dim the lightbulb. To accomplish this, it receives two pieces of information from the Lighting subsystem: a 1-bit signal at every zero-crossing of the AC wave, and the potentiometer reading. When there are people in the room, it sends a pulse back to the dimmer circuitry. This signal is fired at a constant rate: at a specific phase measurement after the zero-crossing of the alternating current. The phase at which the signal is fired is inversely proportional to the potentiometer reading. The importance of the phase is elaborated in the lighting module overview below.

2.3.3 Design Issues & Alternatives

We initially encountered challenges in programming the microcontroller on the PCB. Despite thorough checks for electrical shorts and a review of all component values, we were not able to identify any errors. After reading through online forums and data sheets and consulting with Teaching Assistants we realized the error was due to an incorrectly populated resistor. After removing the resistor we were able to program the microcontroller.

2.4 Lighting Subsystem

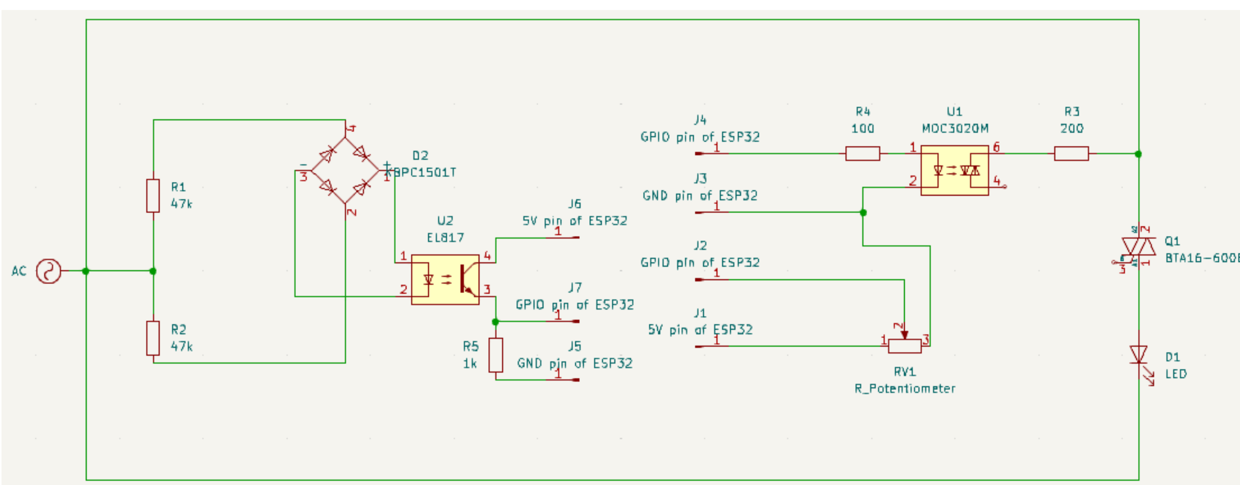

Figure 8: KiCad circuit diagram for the Lighting module [5]

2.4.1 Detecting the Zero-Crossing

For the controller to know when a pulse needs to be sent, it needs to know when the zero-crossing of the AC wave occurs. To do this, we utilize a full bridge rectifier and a photocoupler (left portion of Fig. 8). The rectifier converts the AC wave to a modulating DC wave. By feeding this into a photocoupler, the output voltage is high at all points except when the original wave crosses zero. Thus what results is a trapezoidal wave where each falling edge corresponds to a zero-crossing as shown in Fig. 9.

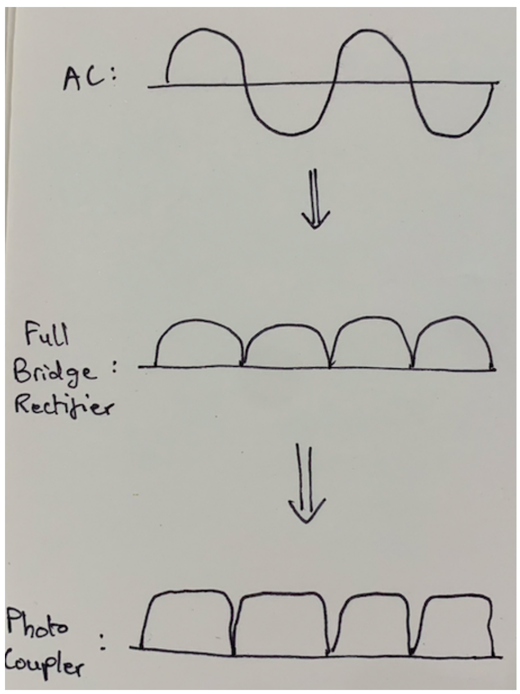

Figure 9: Diagram depicting zero-crossing detection logic

2.4.2 Dimming the Light Bulb

To make the bulb shine less brightly, we modulate the duty cycle of the current going to the bulb via a Triac. The triac is a component that allows current to pass through when it receives a pulse. This current will then pass until it changes direction, at which point the triac will require another pulse. By increasing the potentiometer value, the microcontroller could be told to send the signal at an earlier phase in the AC wave, thereby supplying more power to the bulb and making it shine brighter.

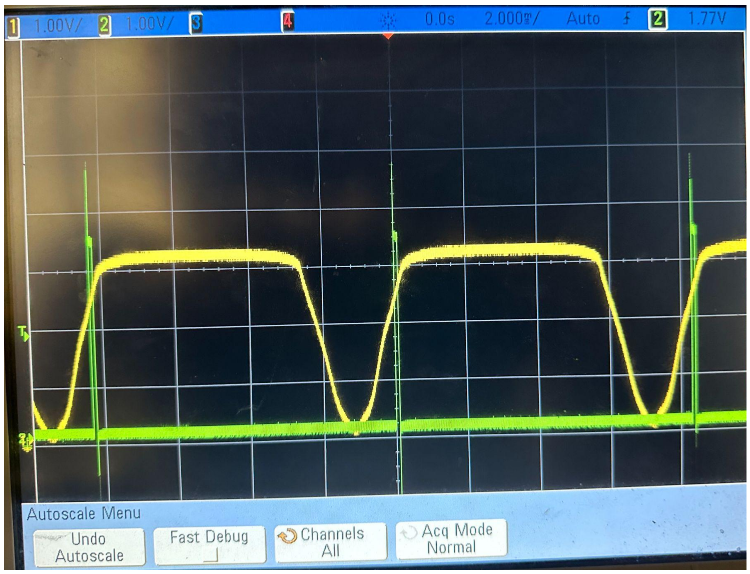

Figure 10: Oscilloscope reading showing pulse generated shortly after zero-crossing (bright bulb)

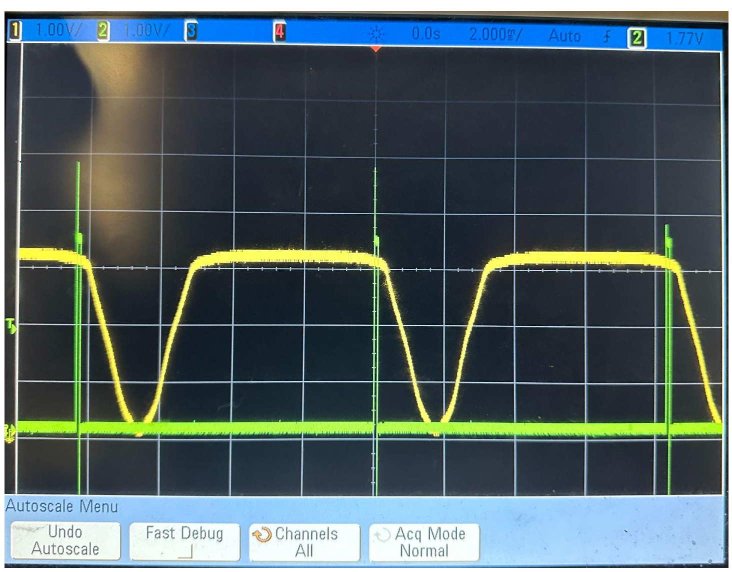

Figure 11: Oscilloscope reading showing pulse generated long after zero-crossing (dim bulb)

2.4.3 Design Issues & Alternatives

Our lighting submodule employs a full bridge rectifier to convert incoming AC current into a DC pulsating wave. When we observed the output signal of the rectifier on the oscilloscope, the signal did not match up with the expected waveform. We then examined our circuit schematic and noticed that the rectifier orientation was incorrect due to a labeling error on the PCB. Upon identifying this discrepancy, we proceeded to re-solder the rectifier in its correct orientation. Following this correction, we successfully observed the expected signal.

2.5 User Interface

2.5.1 Server

Since we want the students to be able to access the room availability data from home, we have our website running on an AWS Linux server. The backend of this website consists of two routes: Update Status and Get Status. Whenever the controller sends room occupancy data via HTTP, it is received by the update route which subsequently modifies the database. Every two seconds, the get status route polls the database for any changes in room status. If there is a change, the corresponding row in the table is modified to reflect it. The final website would look as follows for the user:

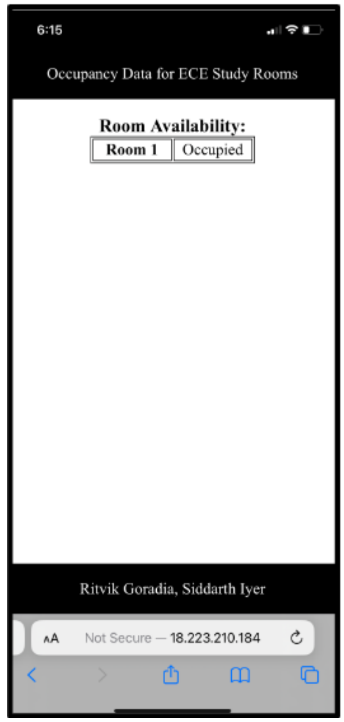

Figure 12: Screenshot of website

2.5.2 Design Issues & Alternatives

We initially decided to use a Google sheet as our online database. However, we realized that Google places a limit on the frequency at which a program can update the Google sheet. This limited the refresh rate of our website. As a result, we decided to switch our database to a JSON file stored on an AWS server instead, where we no longer faced such limitations.

2.6 PCB Design

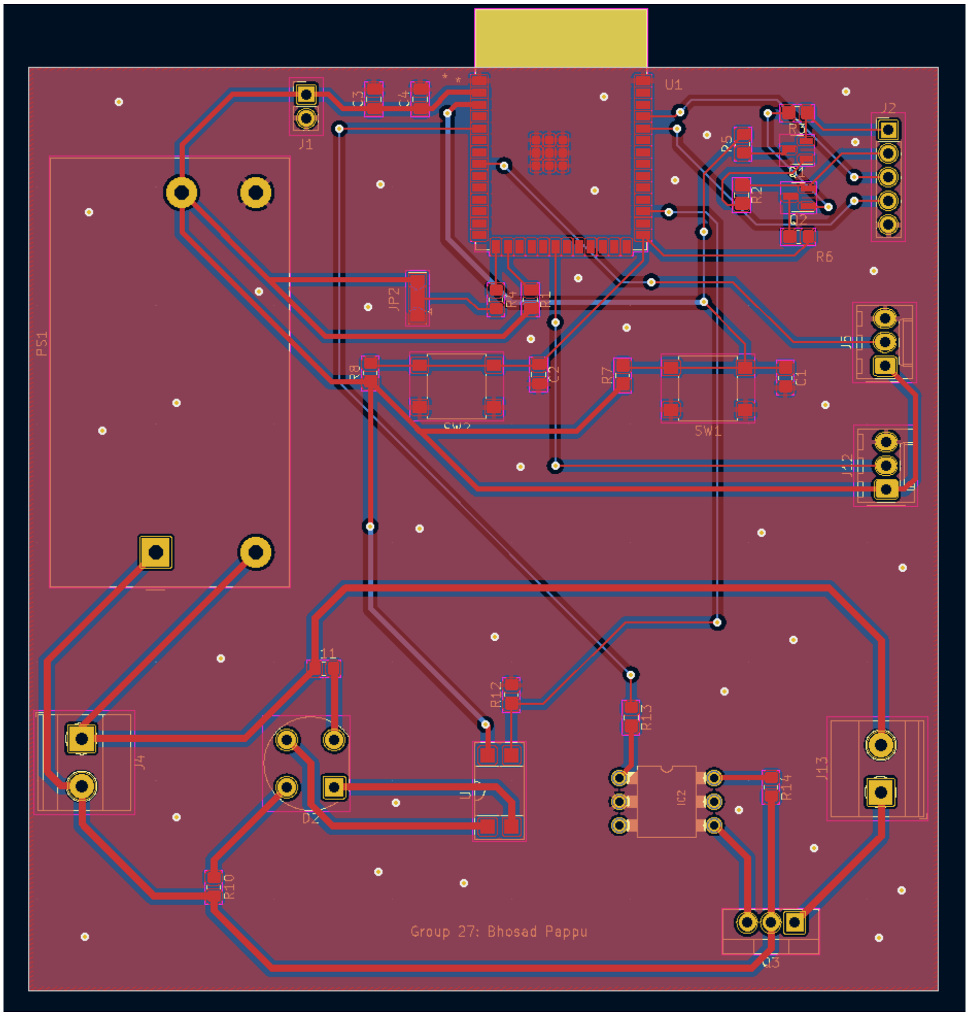

Figure 13: Final PCB layout on KiCad

3. Verification

In this section, we will go over the various subsystem requirements, test plans to verify these requirements, and the results of these tests. A comprehensive list of these requirements and verification is included at the end of the document in Appendix A.

3.1 Sensing

The PIR sensor outputs a 3.3 V 1-bit Digital pulse when motion is detected, and outputs a 0 V digital signal otherwise. To validate the functionality of the sensor, we outlined two requirements.

Requirement 1: The PIR sensor should output Logic Low (0 V) when there is no human present. In order to test this, we connected the output of the PIR sensor to an oscilloscope. We then covered the sensor such that it would not detect any motion and observed a logic low signal on the oscilloscope. Outlined in Table 2 below are the results of this test which show that the requirement was satisfied.

Table 2: Testing results for the first sensor requirement

Test	Low pulse observed?	Test	Low pulse observed?
1	Yes	6	Yes
2	Yes	7	Yes
3	Yes	8	Yes
4	Yes	9	Yes
5	Yes	10	Yes

Requirement 2: The PIR sensor module should output 3.3 V (±0.3 V) digital pulse when a human is detected. To test this, we connected the output of the PIR sensor to an oscilloscope. We then simulated motion over the sensor and observed a logic high signal on the oscilloscope. We also attached a voltmeter to the output of the sensor to measure the outputted voltage. Outlined in Table 3 below are the results of this test which show that the requirement was satisfied

Table 3: Testing results for the second sensor requirement

Test	High pulse observed?	Observed Voltage	Test	High pulse observed?	Observed Voltage
1	Yes	3.293 V	6	Yes	3.294 V
2	Yes	3.293 V	7	Yes	3.294 V
3	Yes	3.293 V	8	Yes	3.293 V
4	Yes	3.294 V	9	Yes	3.293 V
5	Yes	3.293 V	10	Yes	3.293 V

3.2 Controller

The microcontroller receives availability data from the PIR sensor. When motion is detected, the microcontroller sends pulses at a constant frequency of 60 Hz to the lighting module to turn the light on. When no motion is detected, the light must remain off and no pulses are sent. To validate the functionality of the microcontroller, we outlined two requirements.

Requirement 1: Upon receiving a 3.3 V pulse from the sensing submodule, the ESP32 should output a 3.3 V digital signal pulse to the lighting module at a constant rate for 5 minutes (± 10 sec). In order to test this, we connected the output of the GPIO pin that sends pulses to the oscilloscope. We then observed the pulses and ensured that it was being sent a frequency of 60

Hz for a fixed period. Outlined in Table 4 below are the results of this test which show that the requirement was satisfied

Table 4: Testing results for the first controller requirement

Test	Pulse sent at constant rate for 15 s? (Reduced duration for testing purposes)	Test	Pulse sent at constant rate for 15 s? (Reduced duration for testing purposes)
1	Yes	6	Yes
2	Yes	7	Yes
3	Yes	8	Yes
4	Yes	9	Yes
5	Yes	10	Yes

Requirement 2: After not receiving a pulse from the PIR sensor for 5 minutes (± 10 sec), the ESP32 should output a constant 0 V digital signal to the lighting submodule to turn the light off. To test this, we connected the output of the GPIO pin that sends pulses to the oscilloscope. We covered the PIR sensor to ensure no motion was being detected and observed a steady output of 0 V. Outlined in Table 5 below are the results of this test which show that the requirement was satisfied

Table 5: Testing results for the second controller requirement

Test	Stops sending pulse once room is unoccupied?	Test	Stops sending pulse once room is unoccupied?
1	Yes	6	Yes
2	Yes	7	Yes
3	Yes	8	Yes
4	Yes	9	Yes
5	Yes	10	Yes

3.3 Lighting

The lighting submodule is responsible for toggling the light on/off as well as adjusting the brightness of the light. To validate the functionality of the sensor, we outlined two requirements.

Requirement 1: When a 0 V signal is sent to the Diac in the lighting module, the bulb should be unlit. To test this, we sent a 0 V signal to the Diac from the microcontroller and confirmed that the bulb remained off. Outlined in Table 6 below are the results of this test which show that the requirement was satisfied

Table 6: Testing results for the first lighting requirement

Test	Bulb remains unlit	Test	Bulb remains unlit
1	Yes	6	Yes
2	Yes	7	Yes
3	Yes	8	Yes
4	Yes	9	Yes
5	Yes	10	Yes

Requirement 2: Tuning the Potentiometer modifies the brightness of the bulb. To test this, we connected the bulb to the output, gradually tuned the potentiometer from low to high, and ensured that the bulb's brightness went from dim to bright. Outlined in Table 7 below are the results of this test which show that the requirement was satisfied

Table 7: Testing results for the second lighting requirement

Test	Brightness varies from dim to bright	Test	Brightness varies from dim to bright
1	Yes	6	Yes
2	Yes	7	Yes
3	Yes	8	Yes
4	Yes	9	Yes
5	Yes	10	Yes

3.4 User Interface

The user interface receives occupancy data from the microcontroller and accordingly updates a database and the website. To validate that the subsystem is working we outlined one requirement.

Requirement 1: The website update lag should be within 10 seconds. To validate this, we sent an HTTP post request from the microcontroller, started a stopwatch, and stopped it as soon as the website refreshed with the updated information. The graph below indicates our results, and validates our requirements

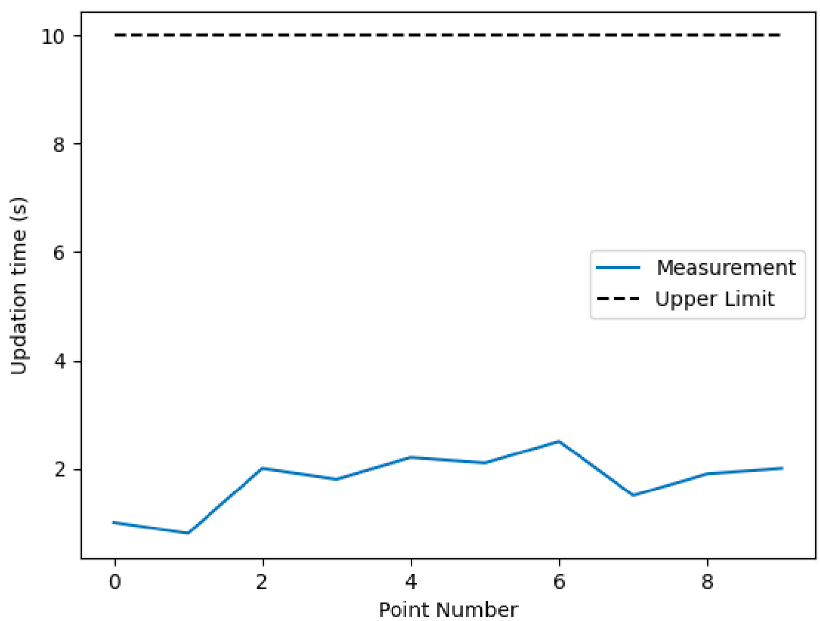

Figure 14: Testing results for the user interface requirement

3.5 Power

The power subsystem is responsible for providing 3.3 V to the various components in the project. To ensure that this subsystem functions as intended, we have outlined one requirement.

Requirement 1: The linear regulator should be able to output 3.3 V (±0.3 V). To validate this requirement, we connected a voltmeter to the output of the linear regulator and observed the readings. Outlined in Table 8 below are the results of this test which show that the requirement was satisfied

Table 8: Testing results for the power subsystem requirement

Test	Observed Voltage	Test	Observed Voltage
1	3.298 V	6	3.296 V
2	3.297 V	7	3.297 V
3	3.297 V	8	3.297 V
4	3.297 V	9	3.298 V
5	3.294 V	10	3.297 V

4. Cost and Schedule

4.1 Cost Analysis

- Labor
  - Salary per hour per person: $40
  - Total number of hours per person on project: 60
  - Team members: 2
  - Total Labor Cost: 40 * 2.5 * 60 * 2 = $12000
- Parts
  - Cost of parts: $85.76 (Full breakdown in Table 10 in Appendix)
  - 5% shipping: $4.29
  - 10% sales tax: $8.58
  - Total cost of parts: $98.63
- Final Cost
  - Labor + Parts: 12000 + 98.63 = $12098.63

4.2 Schedule

In order to complete this project, we have split up the work evenly and created a schedule to complete tasks. We have successfully adhered to this schedule and completed the project on time. Full details of the schedule are available in Table 11 in the appendix.

5. Conclusion

In conclusion, we have successfully prototyped and developed a fully functioning lighting system for the ECEB Study rooms. This system ensures that the real-time availability status of study rooms is accurately reflected on the website. Moreover, the lights automatically turn on or off based on room occupancy and incorporate the added functionality of dimmable lights. We have effectively met all three of our high-level requirements: the website promptly reflects any change in availability within 10 seconds, the room lights activate within 3 seconds of a person entering the room, and users can easily adjust the brightness of the lights. We have successfully satisfied all subsystem-level requirements. In addition, the PCB functioned as intended, and no external modules were required for the system's operation.

During the course of our project development, we encountered several challenges, all of which were ultimately resolved.

One challenge we encountered was difficulty in programming our ESP-32 Microcontroller. Despite following the prescribed steps for programming, we consistently encountered the following error in the Arduino Integrated Development Environment (Arduino IDE): “No serial data received.” We conducted a thorough examination of various PCB solders to ensure the absence of electrical shorts, verified the correctness of our component values, and explored alternative programming methods such as manually pressing the GPIO 0 Strapping and Reset push buttons. However, these efforts did not yield success. To resolve the issue, we conducted extensive research by reviewing online forums and data sheets, and consulting with Teaching Assistants. Ultimately, we identified the root cause as a resistor connected to GPIO 46, which was not intended to be populated. Upon removing this resistor, we successfully programmed our microcontroller.

We encountered another obstacle when one of our essential components, an AC-3.3 V DC converter, did not arrive as expected. This component was critical for powering our low-voltage subsystems. To address this setback, we devised an alternative solution by designing a Linear

Regulator that accepts a 9 V input. By carefully selecting specific resistance values, we ensured the regulator could output the required 3.3 V. We conducted rigorous testing to confirm the proper operation of our linear regulator and its ability to output the correct voltage. We then successfully integrated it into our system and achieved the desired functionality.

In the future, we plan to enhance our website. Presently, there is a potential vulnerability where anyone with the IP address can modify the website. To mitigate this risk, we intend to implement security measures to ensure that unauthorized modifications are prevented. Additionally, we hope to restrict access to the website solely to students and faculty members. To achieve this, we plan on integrating a Single Sign-On (SSO) login system. This will only allow authenticated users to access the website. We also plan on miniaturizing our PCB and refining the overall system's packaging with the goal of transforming our project into a marketable product.

On the surface, our project does not seem to present an ethical problem. However, a problem could arise if this project were expanded to be more accurate using computer vision. This would violate the privacy of the individuals using the study room (taken from the IEEE Code of Ethics [6]). Hence, we will focus our efforts into making the PIR sensing system as accurate as possible. Additionally, this system should easily integrate with the existing power lines in the ECEB study and not pose a risk (electrocution, fire) to the users in the case of failure. Lastly, all team members will be undergoing High-Voltage Training so we will be adequately trained to handle the 120 V coming from the main power line.

References

[1] Pippa, “Pir Vs microwave sensors: Which do you need?,” Green Lighting, https://greenlighting.co.uk/pir-vs-microwave-sensors-need/.

[2] Bombuscreative, “The new Apple iPhone X Silver Color 256GB model with white blank...,” iStock, https://www.istockphoto.com/photo/apple-iphone-x-silver-white-blank-screen-gm892510910-247013254.

[3] “LM317 resistor/voltage calculator,” LM317 Resistor and Voltage Calculator, https://circuitdigest.com/calculators/lm317-resistor-voltage-calculator (accessed Dec. 6, 2023).

[4] LM317 3-terminal adjustable regulator - Texas Instruments India, https://www.ti.com/lit/ds/symlink/lm317.pdf (accessed Dec. 7, 2023).

[5] ElectroNoobs, “Bluetooth controlled TRIAC DIMMER,” Arduino TRIAC dimmer AC bluetooth, http://electronoobs.com/eng_circuitos_tut20.php (accessed Sep. 28, 2023).

[6] “IEEE code of Ethics,” IEEE, https://www.ieee.org/about/corporate/governance/p7-8.html.

Appendix A

Requirement and Verification Table

Table 9: Requirement and Verification table for all subsystems

Requirement	Verification	Verification Status (Y or N)
• The PIR sensor module should output Logic Low (0 V) when there is no human present	• Connect VCC and GND Pins of PIR Sensor to 5 V and GND respectively. Connect the OUT Pin of PIR Sensor to a Logic Analyzer. • Ensure no human is present in front of the sensor. • To pass, the Logic Analyzer Trace should remain at Logic Low for the entire duration. Repeat 10 times and ensure above behavior is observed at least 9 out of 10 times.	Y
• The PIR sensor module should output 3.3 V (±0.3 V) digital pulse when a human is detected	• Connect VCC and GND Pins of PIR sensor to 5 V and GND respectively. Connect the OUT Pin of PIR Sensor to a Logic Analyzer. • Ensure no human is present in front of the sensor and the Logic Analyzer Trace remains at Logic Low. • Now, let a human walk in front of the sensor. • A logic high pulse should be seen on the logic analyzer. Additionally, a Voltmeter should be connected to the OUT Pin in parallel and should read 3.3 V (±0.3 V) upon detecting a human. • Repeat 10 times and ensure above behavior is observed at least 9 out of 10 times.	Y

Requirement	Verification	Verification Status (Y or N)
• Upon receiving a 3.3 V pulse from the sensing submodule, the ESP32 should output a 3.3 V digital signal pulse to the lighting module at a constant rate for 5 minutes (± 10 sec).	• Connect the Output GPIO pin meant to go to the Lighting Circuit to the Logic Analyzer. Also, connect the output of the PIR Sensor to the logic analyzer. While no human is present, ensure that Output of the PIR Sensor is Logic Low. • Now, let a human walk in front of the sensor. Ensure that the pin meant to be connected to the lighting submodule outputs a logic high pulse at a constant rate for 5 minutes (± 10 sec) after the falling edge of the pulse from the PIR sensor is seen. • Repeat 10 times and ensure above behavior is observed at least 9 out of 10 times.	Y
• After not receiving a pulse from the PIR sensor for 5 minutes (± 10 sec), the ESP32 should output a constant 0 V digital signal to the lighting submodule to turn the light off.	• Connect the Output GPIO pin meant to go to the Lighting Submodule to the Logic Analyzer. Also, connect the output of the PIR Sensor to the logic analyzer. • Let a human walk in front of the sensor. • Ensure that after the falling edge of the pulse from the PIR sensor is seen, after 5 minutes (± 10 sec), the output GPIO pin outputs a Logic Low. • Repeat 10 times and ensure above behavior is observed at least 9 out of 10 times.	Y
• When a 0 V signal is sent to the Diac in the lighting module, the bulb should be unlit.	• The input to the Diac should be connected to GND (0 V). The rest of the circuit should be connected as shown in Figure 8.	Y

Requirement	Verification	Verification Status (Y or N)
	● Connect an oscilloscope to the terminal corresponding to the light. ● For this test to pass, the voltage should be 0 V (± 0.3 V) ● Repeat 10 times and ensure above behavior is observed at least 9 out of 10 times.	
● Tuning the Potentiometer modifies the brightness of the bulb.	● Connect 120 V AC to the input of the dimmer circuit. Connect the light bulb to the output terminal ● Tune the potentiometer to the lowest extreme. Observe a dim brightness. ● Gradually tune the potentiometer up and observe the bulb brighten.	Y
● The website update lag should be within 10 seconds (±2 sec).	● As soon as movement is detected by the sensor (indicated by the light turning on), using a stopwatch, the time taken for the room to be updated on the website will be measured. ● This test would be performed 20 times and the mean time should be at most 10 seconds with a variance of 2 seconds.	Y
● Linear regulator should be able to output 3.3 V (±0.3 V).	● Provide a 9 V input to the linear regulator ● Connect a Voltmeter to the output of the linear regulator and ensure voltage reading falls within the required threshold. ● Disconnect the battery and connect it again ● Repeat experiment 10 times and ensure	Y

Requirement	Verification	Verification Status (Y or N)
	that voltage reading falls within the expected threshold at least 9 out of 10 times	

Cost Analysis Table

Table 10: All parts ordered and their cost

Description	**Manufacturer**	**Quantity**	**Total Cost**	**Link**
BS-412 PIR Sensor	Adafruit Industries LLC	3	$5.85	Link
BTA16-600CWRG Triac	STMicroelectronics	2	$4.34	Link
MOC3020 Triac 6DIP	Lite-On Inc.	10	$4.10	Link
IRM-05-3.3	MEAN WELL	4	$10.68	Link
KBP304G Full-bridge Rectifier	Diodes Incorporated	5	$4.15	Link
Resistor Kit	Allecin	1	$7.99	Link
ESP32 Dev Board	DFRobot	2	$14.76	Link

Description	Manufacturer	Quantity	Total Cost	Link
Capacitor Kit	Allecin	1	$9.99	Link
LTV-817 Photocoupler	Lite-On Inc.	10	$3.80	Link
AMS117 Regulator	Bridgold	10	$7.99	Link
7133-1 IC	UMW	10	$3.41	Link
PDB181 Potentiometer	Bourns Inc.	3	$4.56	Link

Schedule

Table 11: Work schedule

Week	Task	Person
September 25th - October 2nd	Order components for development and prototype	Everyone
	Research and finalize on dimmer and relay circuit	Siddarth

Week	Task	Person
	Research on communication b/w ESP and server	Ritvik
October 2nd - October 9th	Finalize overall circuitry and complete first round of PCB Design	Everyone
	Set up Server + Basic Front end and test out communication between ESP 32 and Server using Dev Board	Ritvik
	Test Dimmer and Relay circuitry on a breadboard/perfboard using an oscilloscope	Siddarth
	Bring up sensor firmware and test using a Logic Analyzer	Ritvik/Siddarth
October 9th - October 16th	Complete 2nd iteration of PCB	Everyone
	Complete website front end and back end	Ritvik
	Complete first iteration of ESP 32 software	Siddarth

Week	Task	Person
	Assemble and test 1st iteration of PCB	Everyone
October 16th - October 23rd	Complete final iteration of PCB	Everyone
	Assemble and test 2nd iteration of PCB	Everyone
	Continue development of firmware based on testing	Everyone
October 23rd - October 30th	Start 3D Printing Design for modules	Ritvik
	Assemble and test 3rd iteration of PCB	Everyone
	Finalize penultimate Firmware iteration for ESP 32	Siddarth
October 30th - November 6th	Finalize 3D Design and get it Printed	Ritvik
	Assemble various submodules together and verify all the requirements	Siddarth

Week	Task	Person
November 6th - November 13th	Assemble final module including 3d printed casing	Everyone
	Finalize firmware for ESP32	Everyone
November 13th - November 20th	Address any issues brought up at the mock demo	Everyone